

An Efficient Protocol for Authenticated Key Agreement ¹ ²

Laurie Law
National Security Agency
lelaw@orion.ncsc.mil

Alfred Menezes
University of Waterloo
ajmenez@cacr.math.uwaterloo.ca

Minghua Qu
Certicom Corp.
mqu@certicom.com

Jerry Solinas
National Security Agency
jasolin@orion.ncsc.mil

Scott Vanstone
University of Waterloo
savansto@cacr.math.uwaterloo.ca

¹Technical report CORR 98-05, Dept. of C&O, University of Waterloo, Canada, March 1998.

²Revised: August 28 1998

Abstract

This paper proposes a new and efficient two-pass protocol for authenticated key agreement in the asymmetric (public-key) setting. The protocol is based on Diffie-Hellman key agreement and can be modified to work in an arbitrary finite group and, in particular, elliptic curve groups. Two modifications of this protocol are also presented: a one-pass authenticated key agreement protocol suitable for environments where only one entity is on-line, and a three-pass protocol in which key confirmation is additionally provided. The protocols are currently under consideration for standardization in ANSI X9.42 [2], ANSI X9.63 [4] and IEEE P1363 [18].

Keywords: Diffie-Hellman, authenticated key agreement, key confirmation, elliptic curves.

1 Introduction

Key establishment is the process by which two (or more) entities establish a shared secret key. The key is subsequently used to achieve some cryptographic goal such as confidentiality or data integrity. Broadly speaking, there are two kinds of key establishment protocols: *key transport* protocols in which a key is created by one entity and securely transmitted to the second entity, and *key agreement* protocols in which both parties contribute information which jointly establish the shared secret key. In this paper, we shall only consider key agreement protocols for the asymmetric (public-key) two-entity setting.

Let A and B be two honest entities, i.e., legitimate entities who execute the steps of a protocol correctly. Informally speaking, a key agreement protocol is said to provide *implicit key authentication* (of B to A) if entity A is assured that no other entity aside from a specifically identified second entity B can possibly learn the value of a particular secret key. Note that the property of implicit key authentication does not necessarily mean that A is assured of B actually possessing the key. A key agreement protocol which provides implicit key authentication to both participating entities is called an *authenticated key agreement (AK)* protocol.

Informally speaking, a key agreement protocol is said to provide *key confirmation* (of B to A) if entity A is assured that the second entity B actually has possession of a particular secret key. If both implicit key authentication and key confirmation (of B to A) are provided, then the key establishment protocol is said to provide *explicit key authentication* (of B to A). A key agreement protocol which provides explicit key authentication to both participating entities is called an *authenticated key agreement with key confirmation (AKC)* protocol. For an extensive survey on key establishment, see Chapter 12 of Menezes, van Oorschot and Vanstone [27].

Extreme care must be exercised when separating key confirmation from implicit key authentication. If an AK protocol which does not offer key confirmation is used, then, as pointed out in [10], it is desirable that the agreed key be confirmed prior to cryptographic use. This can be done in a variety of ways. For example, if the key is to be subsequently used to achieve confidentiality, then encryption with the key can begin on some (carefully chosen) known data. Other systems may provide key confirmation during a ‘real-time’ telephone conversation. Separating key confirmation from implicit key authentication is sometimes desirable because it permits flexibility in how a particular implementation chooses to achieve key confirmation, and thus moves the burden of key confirmation from the establishment mechanism to the application.

In this paper, we propose a new and efficient two-pass AK protocol. The protocol is based on Diffie-Hellman key agreement [14], and has many of the desirable security and performance attributes discussed in [10] (see §2). Two modifications of this protocol are also presented: a one-pass AK protocol suitable for environments where only one entity is on-line, and a three-pass protocol in which key confirmation is additionally provided.

The protocols described in this paper establish a *shared secret* K between two entities. A *key derivation function* should then be used to derive a secret key from the shared secret. This

is necessary because K may have some *weak bits* — bits of information about K that can be predicted correctly with non-negligible advantage. One way to derive a secret key from K is to apply a one-way hash function such as SHA-1 [28] to K . With the exception of Protocol 3 in §6.2, this paper does not include key derivation functions in protocol descriptions.

All protocols described in this paper have been described in the setting of the group of points on an elliptic curve defined over a finite field. However, they can all be easily modified to work in any finite group in which the discrete logarithm problem appears intractable. Suitable choices include the multiplicative group of a finite field, subgroups of \mathbb{Z}_n^* where n is a composite integer, and subgroups of \mathbb{Z}_p^* of prime order q . Elliptic curve groups are advantageous because they offer equivalent security as the other groups but with smaller key sizes and faster computation times.

The remainder of the paper is organized as follows. §2 discusses the desirable attributes of AK and AKC protocols. §3 describes the elliptic curve parameters that are common to both entities involved in the protocols, public keys, and methods for validating them. §4 reviews the MTI protocols, and describes some active attacks on them. These attacks influenced the design of the new two-pass AK protocol, which is presented in §5. §6 presents the one-pass variant of the protocol, and the three-pass variant which provides explicit key authentication. Finally, §7 makes concluding remarks.

2 Desirable attributes of AK and AKC protocols

Numerous Diffie-Hellman-based AK and AKC protocols have been proposed over the years; however, many have subsequently been found to have security flaws. The main problems were that appropriate threat models and the goals of secure AK and AKC protocols lacked formal definition. Blake-Wilson, Johnson and Menezes [10], adapting earlier work of Bellare and Rogaway [9] in the symmetric setting, provided a formal model of distributed computing and rigorous definitions of the goals of secure AK and AKC protocols within this model. Concrete AK and AKC protocols were proposed, and proven secure within this framework in the random oracle model [8]. This paper follows their definitions of goals of secure AK and AKC protocols (described informally in §1) and their classification of threats.

A secure protocol should be able to withstand both *passive* attacks (where an adversary attempts to prevent a protocol from achieving its goals by merely observing honest entities carrying out the protocol) and *active* attacks (where an adversary additionally subverts the communications by injecting, deleting, altering or replaying messages). In addition to implicit key authentication and key confirmation, a number of desirable *security attributes* of AK and AKC protocols have been identified (see [10] for a further discussion of these):

1. *known-key security*. Each run of a key agreement protocol between two entities A and B should produce a unique secret key; such keys are called *session keys*. A protocol should still achieve its goal in the face of an adversary who has learned some other session keys.

2. *(perfect) forward secrecy*. If long-term private keys of one or more entities are compromised, the secrecy of previous session keys established by honest entities is not affected.
3. *key-compromise impersonation*. Suppose A 's long-term private key is disclosed. Clearly an adversary that knows this value can now impersonate A , since it is precisely this value that identifies A . However, it may be desirable that this loss does not enable an adversary to impersonate other entities to A .
4. *unknown key-share*. Entity A cannot be coerced into sharing a key with entity B without A 's knowledge, i.e., when A believes the key is shared with some entity $C \neq B$, and B (correctly) believes the key is shared with A .
5. *key control*. Neither entity should be able to force the session key to a preselected value.

Desirable *performance attributes* of AK and AKC protocols include a minimal number of passes (the number of messages exchanged in a run of the protocol), low communication overhead (total number of bits transmitted), and low computation overhead. Other attributes that may be desirable in some circumstances include *role-symmetry* (the messages transmitted between entities have the same structure), *non-interactiveness* (the messages transmitted between the two entities are independent of each other), and the non-reliance on encryption (to meet export requirements), hash functions (since these are notoriously hard to design), and timestamping (since it is difficult to implement securely in practice).

3 Domain parameters and key pair generation

This section describes the elliptic curve parameters that are common to both entities involved in the protocols (i.e., the *domain parameters*), and the key pairs of each entity.

3.1 Domain parameters

The domain parameters for the protocols described in this paper consist of a suitably chosen elliptic curve E defined over a finite field \mathbb{F}_q of characteristic p , and a *base point* $P \in E(\mathbb{F}_q)$. In the remainder of this subsection, we elaborate on what “suitable” parameters are, and outline a procedure for verifying that a given set of parameters meet these requirements.

In order to avoid the Pollard-rho [32] and Pohlig-Hellman [31] algorithms for the elliptic curve discrete logarithm problem (ECDLP), it is necessary that the number of \mathbb{F}_q -rational points on E , denoted $\#E(\mathbb{F}_q)$, be divisible by a sufficiently large prime n . As of this writing, it is commonly recommended that $n > 2^{160}$ (see [3, 4]). Having fixed an underlying field \mathbb{F}_q , n should be selected to be as large as possible, i.e., one should have $n \approx q$, so $\#E(\mathbb{F}_q)$ is *almost prime*. In the remainder of this paper, we shall assume that $n > 2^{160}$ and that $n > 4\sqrt{q}$. By Hasse's Theorem,

$$(\sqrt{q} - 1)^2 \leq \#E(\mathbb{F}_q) \leq (\sqrt{q} + 1)^2.$$

Hence $n > 4\sqrt{q}$ implies that n^2 does not divide $\#E(\mathbb{F}_q)$, and thus $E(\mathbb{F}_q)$ has a unique subgroup of order n . Also, since

$$(\sqrt{q} + 1)^2 - (\sqrt{q} - 1)^2 = 4\sqrt{q},$$

there is a unique integer h such that

$$q + 1 - 2\sqrt{q} \leq nh \leq q + 1 + 2\sqrt{q},$$

namely $h = \lfloor (\sqrt{q} + 1)^2/n \rfloor$; hence $\#E(\mathbb{F}_q) = nh$. To guard against potential small subgroup attacks (see §4.1), the point P should have order n .

Some further precautions should be exercised when selecting the elliptic curve. To avoid the reduction algorithms of Menezes, Okamoto and Vanstone [24] and Frey and Rück [16], the curve should be non-supersingular (i.e., p should not divide $(q + 1 - \#E(\mathbb{F}_q))$). More generally, one should verify that n does not divide $q^k - 1$ for all $1 \leq k \leq C$, where C is large enough so that it is computationally infeasible to find discrete logarithms in \mathbb{F}_{q^C} ($C = 20$ suffices in practice [3]). To avoid the attack of Semaev [34], Smart [35], and Satoh and Araki [33] on \mathbb{F}_q -anomalous curves, the curve should not be \mathbb{F}_q -anomalous (i.e., $\#E(\mathbb{F}_q) \neq q$).

A prudent way to guard against these attacks, and similar attacks against special classes of curves that may be discovered in the future, is to select the elliptic curve E at random subject to the condition that $\#E(\mathbb{F}_q)$ is divisible by a large prime — the probability that a random curve succumbs to these special-purpose attacks is negligible. A curve can be selected *verifiably at random* by choosing the coefficients of the defining elliptic curve equation as the outputs of a one-way function such as SHA-1 according to some pre-specified procedure. A procedure for accomplishing this, similar in spirit to the method given in FIPS 186 [29] for selecting DSA primes verifiably at random, is described in ANSI X9.62 [3].

To summarize, domain parameters are comprised of:

1. a field size q , where q is a prime power (in practice, either $q = p$, an odd prime, or $q = 2^m$);
2. an indication FR (*field representation*) of the representation used for the elements of \mathbb{F}_q ;
3. two field elements a and b in \mathbb{F}_q which define the equation of the elliptic curve E over \mathbb{F}_q (i.e., $y^2 = x^3 + ax + b$ in the case $p > 3$, and $y^2 + xy = x^3 + ax^2 + b$ in the case $p = 2$);
4. two field elements x_P and y_P in \mathbb{F}_q which define a finite point $P = (x_P, y_P)$ of prime order in $E(\mathbb{F}_q)$ (since P is described by two field elements, this implies that $P \neq \mathcal{O}$, where \mathcal{O} denotes the point at infinity);
5. the order n of the point P ; and
6. the cofactor $h = \#E(\mathbb{F}_q)/n$.

DOMAIN PARAMETER VALIDATION. A set of domain parameters $(q, \text{FR}, a, b, P = (x_P, y_P), n, h)$ can be verified to meet the above requirements as follows. This process is called *domain parameter validation*.

1. Verify that q is a prime power.
2. Verify that FR is a valid field representation.
3. Verify that a , b , x_P and y_P are elements of \mathbb{F}_q (i.e., verify that they are of the proper format for elements of \mathbb{F}_q).
4. Verify that a and b define a (non-singular) elliptic curve over \mathbb{F}_q (i.e., $4a^3 + 27b^2 \neq 0$ in the case $p > 3$, and $b \neq 0$ in the case $p = 2$).
5. Verify that P satisfies the defining equation of E (and that $P \neq \mathcal{O}$).
6. Verify that $n > 4\sqrt{q}$, that n is prime, and that n is sufficiently large (e.g., $n > 2^{160}$).
7. Verify that $nP = \mathcal{O}$.
8. Compute $h' = \lfloor (\sqrt{q} + 1)^2 / n \rfloor$, and verify that $h = h'$.
9. To ensure protection against known attacks on special classes of elliptic curves, verify that n does not divide $q^k - 1$ for all $1 \leq k \leq 20$, and that $n \neq q$.

3.2 Key pair generation

Given a valid set of domain parameters $(q, \text{FR}, a, b, P, n, h)$, an entity A 's *private key* is an integer d selected at random from the interval $[1, n - 1]$. A 's *public key* is the elliptic curve point $Q = dP$. The *key pair* is (Q, d) . Each run of a key agreement protocol between two entities A and B should produce a unique shared secret key. Hence, for the protocols described in this paper, each entity has two public keys: a *static* or *long-term* public key, and an *ephemeral* or *short-term* public key. The static public key is bound to the entity for a certain period of time, typically through the use of certificates. A new ephemeral public key is generated for each run of the protocol. A 's static key pair is denoted (W_A, w_A) , while A 's ephemeral key pair is denoted (R_A, r_A) .

For the remainder of this paper, we will assume that static public keys are exchanged via certificates. Cert_A denotes A 's public-key certificate, containing a string of information that uniquely identifies A (such as A 's name and address), her static public key W_A , and a certifying authority CA's signature over this information. To avoid a potential unknown key-share attack (see §4.2), the CA should verify that A possesses the private key w_A corresponding to her static public key W_A . Other information may be included in the data portion of the certificate, including the domain parameters if these are not known from context. Any other entity B can use his authentic copy of the CA's public key to verify A 's certificate, thereby obtaining an authentic copy of A 's static public key.

PUBLIC-KEY VALIDATION. Before using an entity's purported public key Q , it is prudent to verify that it possesses the arithmetic properties it is supposed to — namely that Q be a finite point in the subgroup generated by P . This process is called *public-key validation* [19]. Given a valid set of domain parameters $(q, \text{FR}, a, b, P, n, h)$, a purported public key $Q = (x_Q, y_Q)$ can be validated by verifying that:

1. Q is not equal to \mathcal{O} ;
2. x_Q and y_Q are elements in the field \mathbb{F}_q (i.e., they are of the proper format for elements of \mathbb{F}_q);
3. Q satisfies the defining equation of E ; and
4. $nQ = \mathcal{O}$.

EMBEDDED PUBLIC-KEY VALIDATION. The computationally expensive operation in public-key validation is the scalar multiplication in step 4. For static public keys, the validation could be done once and for all by the CA. However, since a new ephemeral key is generated for each run of the protocol, validation of the ephemeral key places a significant burden on the entity performing the validation. To reduce this burden, step 4 can be omitted during key validation. Instead, the protocols proposed in §5 and §6 ensure that the shared secret K generated is a finite point in the subgroup generated by P . To summarize, given a valid set of domain parameters $(q, \text{FR}, a, b, P, n, h)$, *embedded public-key validation* of a purported public key $Q = (x_Q, y_Q)$ is accomplished by verifying that:

1. Q is not equal to \mathcal{O} ;
2. x_Q and y_Q are elements in the field \mathbb{F}_q (i.e., they are of the proper format for elements of \mathbb{F}_q); and
3. Q satisfies the defining equation of E .

4 The MTI key agreement protocols

The MTI/A0 and MTI/C0 key agreement protocols described here are special cases of the three infinite families of key agreement protocols proposed by Matsumoto, Takashima and Imai [23] in 1986. They were designed to provide implicit key authentication, and do not provide key confirmation. Closely related protocols are KEA [30] and those proposed by Goss [17] and Yacobi [37], the latter operating in the ring of integers modulo a composite integer. Yacobi proved that his protocol is secure against certain types of known-key attacks by a passive adversary (provided that the composite modulus Diffie-Hellman problem is intractable). However, Desmedt and Burmester [13] pointed out that the security is only heuristic under known-key attack by an active adversary.

This section illustrates that the MTI/A0 and MTI/C0 families of protocols are vulnerable to several active attacks. The small subgroup attack presented in §4.1 illustrates that the MTI/C0 protocol (as originally described) does not provide implicit key authentication. The unknown key-share attack presented in §4.2 illustrates that the MTI/A0 protocol does not possess the unknown key-share attribute in some circumstances. Other active attacks on AK protocols are discussed by Diffie, van Oorschot and Wiener [15], Burmester [11], Just and Vaudenay [20], and Lim and Lee [22].

4.1 Small subgroup attack

The small subgroup attack was first pointed out by Vanstone [26]; see also van Oorschot and Wiener [36], Anderson and Vaudenay [1], and Lim and Lee [22]. The attack illustrates that authenticating and validating the static and ephemeral keys is a prudent, and sometimes essential, measure to take in Diffie-Hellman AK protocols. We illustrate the small subgroup attack on the MTI/C0 protocol. The protocol assumes that A and B a priori possess authentic copies of each other's static public keys.

MTI/C0 AK PROTOCOL.

1. A generates a random integer r_A , $1 \leq r_A \leq n - 1$, computes the point $T_A = r_A W_B$, and sends T_A to B .
2. B generates a random integer r_B , $1 \leq r_B \leq n - 1$, computes the point $T_B = r_B W_A$, and sends T_B to A .
3. A computes $K = w_A^{-1} r_A T_B = r_A r_B P$.
4. B computes $K = w_B^{-1} r_B T_A = r_A r_B P$.
5. The shared secret is the point K .

The small subgroup attack can be launched if the order n of the base point P is not prime; say, $n = mt$ where $t > 1$ is small¹. The attack forces the shared secret to be one of a small and known subset of points.

A SMALL SUBGROUP ATTACK ON THE MTI/C0 PROTOCOL.

1. E intercepts A 's message T_A and replaces it with mT_A .
2. E intercepts B 's message T_B and replaces it with mT_B .
3. A computes $K = w_A^{-1} r_A (mT_B) = r_A r_B (mP)$.
4. B computes $K = w_B^{-1} r_B (mT_A) = r_A r_B (mP)$.

K lies in the subgroup of order t of the group generated by P , and hence it takes on one of only t possible values. Since the value of K can be correctly guessed by E with high probability, this shows that the MTI/C0 protocol does not provide implicit key authentication. The effects of this can be especially drastic because a subsequent key confirmation phase may fail to detect this attack — E may be able to correctly compute the messages required for key confirmation. For example, E may be able to compute on behalf of A the message authentication code (MAC) under K of the message consisting of the identities of A and B , the ephemeral point mT_A purportedly sent by A , and the point T_B sent by B .

The small subgroup attack in MTI/C0 can be prevented, for example, by mandating use of a base point P of prime order, and requiring that both A and B perform key validation on the ephemeral public keys they receive.

¹In this case, static private keys w should be selected subject to the condition $\gcd(w, n) = 1$.

4.2 Unknown key-share attack

The active attack on the MTI/A0 protocol described in this subsection was first pointed out by Menezes, Qu and Vanstone [25]. The attack, and the many variants of it, illustrate that it is prudent to require an entity A to prove possession of the private key corresponding to its (static) public key to a CA before the CA certifies the public key as belonging to A . This proof of possession can be accomplished by zero-knowledge techniques; for example, see Chaum, Evertse and van de Graaf [12].

MTI/A0 AK PROTOCOL.

1. A generates a random integer r_A , $1 \leq r_A \leq n - 1$, computes the point $R_A = r_AP$, and sends (R_A, Cert_A) to B .
2. B generates a random integer r_B , $1 \leq r_B \leq n - 1$, computes the point $R_B = r_BP$, and sends (R_B, Cert_B) to A .
3. A computes $K = w_AR_B + r_AW_B = (w_Ar_B + w_Br_A)P$.
4. B computes $K = w_BR_A + r_BW_A = (w_Ar_B + w_Br_A)P$.
5. The shared secret is the point K .

In one variant of the unknown key-share attack, the adversary E wishes to have messages sent from A to B identified as having originated from E herself. To accomplish this, E selects an integer e , $1 \leq e \leq n - 1$, computes $W_E = eW_A = ew_AP$, and gets this certified as her public key. Notice that E does not know the logical private key $w_E = ew_A \bmod n$ which corresponds to her public key (assuming, of course, that the discrete logarithm problem in $E(\mathbb{F}_q)$ is intractable), although she knows e .

AN UNKNOWN KEY-SHARE ATTACK ON THE MTI/A0 PROTOCOL.

1. E intercepts A 's message (R_A, Cert_A) and replaces it with (R_A, Cert_E) .
2. B sends (R_B, Cert_B) to E , who then forwards (eR_B, Cert_B) to A .
3. A computes $K = w_A(eR_B) + r_AW_B = (ew_Ar_B + w_Br_A)P$.
4. B computes $K = w_BR_A + r_BW_E = (ew_Ar_B + w_Br_A)P$.

A and B now share the secret K even though B believes he shares the secret with E . Note that E does not learn the value of K . Hence, the attack illustrates that the MTI/A0 protocol does not possess the unknown key-share attribute.

A hypothetical scenario where the attack may be launched successfully is the following; this scenario was first described by Diffie, van Oorschot and Wiener [15]. Suppose that B is a bank branch and A is an account holder. Certificates are issued by the bank headquarters and within each certificate is the account information of the holder. Suppose that the protocol for electronic deposit of funds is to exchange a key with a bank branch via a mutually authenticated key agreement. Once B has authenticated the transmitting entity, encrypted funds are deposited

to the account number in the certificate. Suppose that no further authentication is done in the encrypted deposit message (which might be the case to save bandwidth). If the attack mentioned above is successfully launched then the deposit will be made to E 's account instead of A 's account.

5 The new authenticated key agreement protocol

In this section, the two-pass AK protocol is described. The following notation is used in §5 and §6. f denotes the bitlength of n , the prime order of the base point P ; i.e., $f = \lceil \log_2 n \rceil + 1$. If Q is a finite elliptic curve point, then \overline{Q} is defined as follows. Let x be the x -coordinate of Q , and let \overline{x} be the integer obtained from the binary representation of x . (The value of \overline{x} will depend on the representation chosen for the elements of the field \mathbb{F}_q .) Then \overline{Q} is defined to be the integer $(\overline{x} \bmod 2^{\lceil f/2 \rceil}) + 2^{\lceil f/2 \rceil}$. Observe that $(\overline{Q} \bmod n) \neq 0$.

5.1 Protocol description

We now describe the two-pass AK protocol (Protocol 1) which is an optimization and refinement of a protocol first described by Menezes, Qu and Vanstone [25]. It is depicted in Figure 1. In this and subsequent Figures, $A^{(w_A, W_A)}$ denotes that A 's static private key and static public key are w_A and W_A , respectively. Domain parameters and static keys are set up and validated as described in §3. If A and B do not a priori possess authentic copies of each other's static public keys, then certificates should be included in the flows.

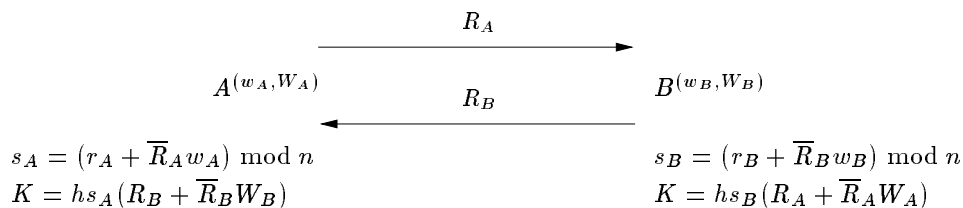


Figure 1: Two-pass AK protocol (Protocol 1)

PROTOCOL 1.

1. A generates a random integer r_A , $1 \leq r_A \leq n - 1$, computes the point $R_A = r_A P$, and sends this to B .
2. B generates a random integer r_B , $1 \leq r_B \leq n - 1$, computes the point $R_B = r_B P$, and sends this to A .
3. A does an embedded key validation of R_B (see §3.2). If the validation fails, then A terminates the protocol run with failure. Otherwise, A computes

$$s_A = (r_A + \overline{R}_A w_A) \bmod n \quad (1)$$

and

$$K = hs_A(R_B + \overline{R}_B W_B). \quad (2)$$

If $K = \mathcal{O}$, then A terminates the protocol run with failure.

4. B does an embedded key validation of R_A . If the validation fails, then B terminates the protocol run with failure. Otherwise, B computes

$$s_B = (r_B + \overline{R}_B w_B) \bmod n \quad (3)$$

and

$$K = hs_B(R_A + \overline{R}_A W_A). \quad (4)$$

If $K = \mathcal{O}$, then B terminates the protocol run with failure.

5. The shared secret is the point K .

5.2 Security notes and rationale

Although the security of Protocol 1 has not been formally proven in a model of distributed computing, heuristic arguments suggest that Protocol 1 provides mutual implicit key authentication. In addition, Protocol 1 appears to have the security attributes of known-key security, forward secrecy, key-compromise impersonation, and key control that were listed in §2. Another security attribute of Protocol 1 is that compromise of the ephemeral private keys (r_A and r_B) reveals neither the static private keys (w_A and w_B) nor the shared secret K .

Kaliski [21] has recently observed that Protocol 1 does not possess the unknown key-share attribute. This is demonstrated by the following on-line attack. The adversary E intercepts A 's ephemeral public key R_A intended for B , and computes $R_E = R_A + \overline{R}_A W_A - P$, $w_E = (\overline{R}_E)^{-1} \bmod n$, and $W_E = w_E P$. E then gets W_E certified as her static public key (note that E knows the corresponding private key w_E), and transmits R_E to B . B responds by sending R_B to E , which E forwards to A . Both A and B compute the same secret K , however B mistakenly believes that he shares K with E . We emphasize that lack of the unknown key-share attribute does not contradict the fundamental goal of mutual implicit key authentication — by definition the provision of implicit key authentication is only considered in the case where B engages in the protocol with an honest entity (which E isn't). If an application using Protocol 1 is concerned with the lack of the unknown key-share attribute under such on-line attacks, then appropriate key confirmation should be added, for example as specified in Protocol 3.

Protocol 1 can be viewed as a direct extension of the ordinary (unauthenticated) Diffie-Hellman key agreement protocol. The quantities s_A and s_B serve as *implicit signatures* for A 's ephemeral public key R_A and B 's ephemeral public key R_B , respectively. The shared secret is

$$K = hs_A s_B P = h(r_A r_B + r_A w_B \overline{R}_B + r_B w_A \overline{R}_A + w_A w_B \overline{R}_A \overline{R}_B) P, \quad (5)$$

rather than $r_A r_B P$ as would be the case with ordinary Diffie-Hellman.

The expression for \overline{R}_A uses only half the bits of the x -coordinate of R_A . This was done in order to increase the efficiency of computing K because the scalar multiplication $\overline{R}_A W_A$ in (4) can be done in half the time of a full scalar multiplication. The modification does not appear to affect the security of the protocol. The definition of \overline{R}_A implies that $\overline{R}_A \neq 0$; this ensures that the contribution of the static private key w_A is not being cancelled in the formation of s_A in (1).

Multiplication by h in (2) and (4) ensures that the shared secret K (see equation (5)) is a point in the subgroup of order n in $E(\mathbb{F}_q)$. The check $K = \mathcal{O}$ ensures that K is a finite point.

If Protocol 1 is used to agree upon a k -bit key for subsequent use in a symmetric-key block cipher, then it is recommended that the elliptic curve be chosen so that $n > 2^{2k}$.

Protocol 1 has all the desirable performance attributes listed in §2. From A 's point of view, the dominant computational steps in a run of Protocol 1 are the scalar multiplications $r_A P$, $\overline{R}_B W_B$, and $h s_A (R_B + \overline{R}_B W_B)$. Hence the work required by each entity is 2.5 (full) scalar multiplications. Since $r_A P$ can be computed off-line by A , the on-line work required by each entity is only 1.5 scalar multiplications. In addition, the protocol has low communication overhead, is role-symmetric, non-interactive, and does not use encryption or timestamping. While a hash function may be used in the key derivation function (to derive a session key from the shared secret K), the security of Protocol 1 appears to be less reliant on the cryptographic strength of the hash function than some other AK protocols (such as Protocol 3 in [10]). In particular, the requirement that the key derivation function be preimage resistant appears unnecessary. Non-reliance on hash functions is advantageous because history has shown that secure hash functions are difficult to design.

6 Related protocols

This section presents two related protocols: a one-pass AK protocol (Protocol 2), and a three-pass AKC protocol (Protocol 3).

6.1 One-pass authenticated key agreement protocol

The purpose of the one-pass AK protocol (Protocol 2) is for entities A and B to establish a shared secret by only having to transmit one message from A to B . This can be useful in applications where only one entity is on-line, such as secure email and store-and-forward. Protocol 2 is depicted in Figure 2. It assumes that A a priori has an authentic copy of B 's static public key. Domain parameters and static keys are set up and validated as described in §3.

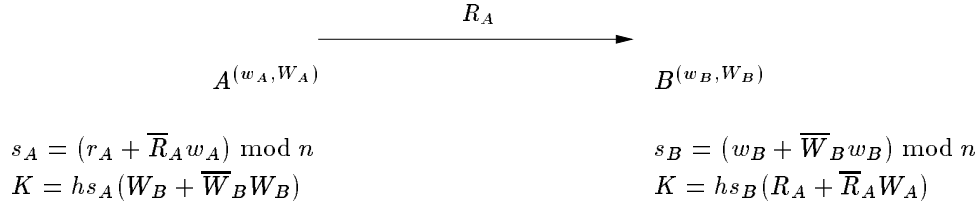


Figure 2: One-pass AK protocol (Protocol 2)

PROTOCOL 2.

1. A generates a random integer r_A , $1 \leq r_A \leq n - 1$, computes the point $R_A = r_A P$, and sends this to B .
2. A computes $s_A = (r_A + \overline{R}_A w_A) \bmod n$ and $K = hs_A(W_B + \overline{W}_B W_B)$. If $K = \mathcal{O}$, then A terminates the protocol run with failure.
3. B does an embedded key validation of R_A (see §3.2). If the validation fails, then B terminates the protocol run with failure. Otherwise, B computes $s_B = (w_B + \overline{W}_B w_B) \bmod n$ and $K = hs_B(R_A + \overline{R}_A W_A)$. If $K = \mathcal{O}$, then B terminates the protocol run with failure.
4. The shared secret is the point K .

Heuristic arguments suggest that Protocol 2 offers mutual implicit key authentication. The main security drawback of Protocol 2 is that there is no known-key security and forward secrecy since entity B does not contribute a random per-message component.. Of course, this will be the case with any one-pass key agreement protocol.

6.2 Authenticated key agreement with key confirmation protocol

We now describe the AKC variant of Protocol 1. It is depicted in Figure 3. Domain parameters and static keys are set up and validated as described in §3. Here, MAC is a message authentication code algorithm and is used to provide key confirmation. For examples of provably secure and practical MAC algorithms, see Bellare, Canetti and Krawczyk [5], Bellare, Guerin and Rogaway [6], and Bellare, Kilian and Rogaway [7]. \mathcal{H}_1 and \mathcal{H}_2 are (independent) key derivation functions. Practical instantiations of \mathcal{H}_1 and \mathcal{H}_2 include $\mathcal{H}_1(z) = \text{SHA-1}(01, z)$ and $\mathcal{H}_2(z) = \text{SHA-1}(10, z)$.

PROTOCOL 3.

1. A generates a random integer r_A , $1 \leq r_A \leq n - 1$, computes the point $R_A = r_A P$, and sends this to B .
2. 2.1 B does an embedded key validation of R_A (see §3.2). If the validation fails, then B terminates the protocol run with failure.
 2.2 Otherwise, B generates a random integer r_B , $1 \leq r_B \leq n - 1$, computes the point $R_B = r_B P$.

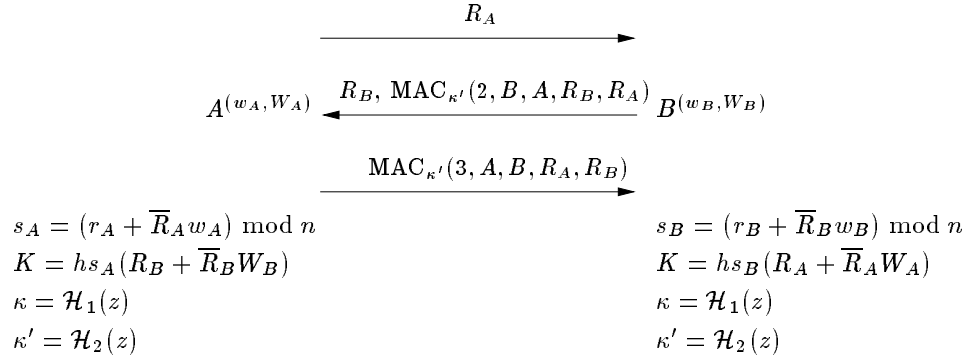


Figure 3: Three-pass AKC protocol (Protocol 3)

- 2.3 B computes $s_B = (r_B + \overline{R_B}w_B) \bmod n$ and $K = hs_B(R_A + \overline{R_A}W_A)$. If $K = \mathcal{O}$, then B terminates the protocol run with failure. The shared secret is K .
- 2.4 B uses the x -coordinate z of the point K to compute two shared keys $\kappa = \mathcal{H}_1(z)$ and $\kappa' = \mathcal{H}_2(z)$.
- 2.5 B computes $\text{MAC}_{\kappa'}(2, B, A, R_B, R_A)$ and sends this together with R_B to A .
3. 3.1 A does an embedded key validation of R_B . If the validation fails, then A terminates the protocol run with failure.
- 3.2 Otherwise, A computes $s_A = (r_A + \overline{R_A}w_A) \bmod n$ and $K = hs_A(R_B + \overline{R_B}W_B)$. If $K = \mathcal{O}$, then A terminates the protocol run with failure.
- 3.3 A uses the x -coordinate z of the point K to compute two shared keys $\kappa = \mathcal{H}_1(z)$ and $\kappa' = \mathcal{H}_2(z)$.
- 3.4 A computes $\text{MAC}_{\kappa'}(2, B, A, R_B, R_A)$ and verifies that this equals what was sent by B .
- 3.5 A computes $\text{MAC}_{\kappa'}(3, A, B, R_A, R_B)$ and sends this to B .
4. B computes $\text{MAC}_{\kappa'}(3, A, B, R_A, R_B)$ and verifies that this equals what was sent by A .
5. The session key is κ .

AKC Protocol 3 is derived from AK Protocol 1 by adding key confirmation to the latter. This is done in exactly the same way AKC Protocol 2 of [10] was derived from AK Protocol 3 of [10]. Protocol 2 of [10] was formally proven to be a secure AKC protocol. Heuristic arguments suggest that Protocol 3 of this paper is a secure AKC protocol, and in addition has all the desirable security attributes listed in §2.

7 Concluding remarks

The paper has presented new AK and AKC protocols which possess many desirable security attributes, are extremely efficient, and appear to place less burden on the security of the key

derivation function than other proposals. Neither of the protocols proposed have been formally proven to possess the claimed levels of security, but heuristic arguments suggest that this is the case. It is hoped that the protocols, or appropriate modifications of them, can, under plausible assumptions, be proven secure in the model of distributed computing introduced in [10].

Acknowledgements

The authors would like to thank Simon Blake-Wilson and Don Johnson for their comments on earlier drafts of this paper. Ms. Law and Dr. Solinas would like to acknowledge the contributions of their colleagues at the National Security Agency.

References

- [1] R. Anderson and S. Vaudenay, "Minding your p 's and q 's", *Advances in Cryptology – Asiacrypt '96*, Lecture Notes in Computer Science, **1163**, Springer-Verlag, 1996, 26-35.
- [2] ANSI X9.42, *Agreement of Symmetric Algorithm Keys Using Diffie-Hellman*, working draft, May 1998.
- [3] ANSI X9.62, *The Elliptic Curve Digital Signature Algorithm (ECDSA)*, working draft, June 1998.
- [4] ANSI X9.63, *Elliptic Curve Key Agreement and Key Transport Protocols*, working draft, July 1998.
- [5] M. Bellare, R. Canetti and H. Krawczyk, "Keying hash functions for message authentication", *Advances in Cryptology – Crypto '96*, Lecture Notes in Computer Science, **1109**, Springer-Verlag, 1996, 1-15.
- [6] M. Bellare, R. Guerin, and P. Rogaway, "XOR MACs: New methods for message authentication using finite pseudorandom functions", *Advances in Cryptology – Crypto '95*, Lecture Notes in Computer Science, **963**, Springer-Verlag, 1995, 15-28.
- [7] M. Bellare, J. Kilian, and P. Rogaway, "The security of cipher block chaining", *Advances in Cryptology – Crypto '94*, Lecture Notes in Computer Science, **839**, Springer-Verlag, 1994, 341-358.
- [8] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols", *1st ACM Conference on Computer and Communications Security*, 1993, 62-73.
- [9] M. Bellare and P. Rogaway, "Entity authentication and key distribution", *Advances in Cryptology – Crypto '93*, Lecture Notes in Computer Science, **773**, Springer-Verlag, 1994, 232-249.

-
- [10] S. Blake-Wilson, D. Johnson and A. Menezes, “Key agreement protocols and their security analysis”, *Proceedings of the sixth IMA International Conference on u Cryptography and Coding*, Lecture Notes in Computer Science, **1355**, Springer-Verlag, 1997, 30-45.
- [11] M. Burmester, “On the risk of opening distributed keys”, *Advances in Cryptology – Crypto ’94*, Lecture Notes in Computer Science, **839**, Springer-Verlag, 1994, 308-317.
- [12] D. Chaum, J.-H. Evertse and J. van de Graaf, “An improved protocol for demonstrating possession of discrete logarithms and some generalizations”, *Advances in Cryptology – Eurocrypt ’87*, Lecture Notes in Computer Science, **304**, Springer-Verlag, 1988, 127-141.
- [13] Y. Desmedt and M. Burmester, “Towards practical ‘proven secure’ authenticated key distribution”, *1st ACM Conference on Computer and Communications Security*, 1993, 228–231.
- [14] W. Diffie and M. Hellman, “New directions in cryptography”, *IEEE Transactions on Information Theory*, **22** (1976), 644-654.
- [15] W. Diffie, P. van Oorschot and M. Wiener, “Authentication and authenticated key exchanges”, *Designs, Codes and Cryptography*, **2** (1992), 107-125.
- [16] G. Frey and H. Rück, “A remark concerning m -divisibility and the discrete logarithm in the divisor class group of curves”, *Mathematics of Computation*, **62** (1994), 865-874.
- [17] K.C. Goss, “Cryptographic method and apparatus for public key exchange with authentication”, U.S. patent 4,956,865, September 11 1990.
- [18] IEEE P1363, *Standard Specifications for Public-Key Cryptography*, working draft, July 1998.
- [19] D. Johnson, Contribution to ANSI X9F1 working group, 1997.
- [20] M. Just and S. Vaudenay, “Authenticated multi-party key agreement”, *Advances in Cryptology – Asiacrypt ’96*, Lecture Notes in Computer Science, **1163**, Springer-Verlag, 1996, 36-49.
- [21] B. Kaliski, Contribution to ANSI X9F1 and IEEE P1363 working groups, June 1998.
- [22] C. Lim and P. Lee, “A key recovery attack on discrete log-based schemes using a prime order subgroup”, *Advances in Cryptology – Crypto ’97*, Lecture Notes in Computer Science, **1294**, Springer-Verlag, 1997, 249-263.
- [23] T. Matsumoto, Y. Takashima and H. Imai, “On seeking smart public-key distribution systems”, *The Transactions of the IECE of Japan*, **E69** (1986), 99-106.
- [24] A. Menezes, T. Okamoto and S. Vanstone, “Reducing elliptic curve logarithms to logarithms in a finite field”, *IEEE Transactions on Information Theory*, **39** (1993), 1639-1646.

-
- [25] A. Menezes, M. Qu and S. Vanstone, “Some new key agreement protocols providing mutual implicit authentication”, *Workshop on Selected Areas in Cryptography (SAC '95)*, 22-32, 1995.
- [26] A. Menezes, M. Qu and S. Vanstone, “Key agreement and the need for authentication”, presentation at PKC '95, Toronto, Canada, November 1995.
- [27] A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
- [28] National Institute of Standards and Technology, “Secure Hash Standard (SHS)”, FIPS Publication 180-1, April 1995.
- [29] National Institute of Standards and Technology, “Digital signature standard”, FIPS Publication 186, 1994.
- [30] National Security Agency, “SKIPJACK and KEA algorithm specification”, Version 2.0, May 29 1998.
- [31] S. Pohlig and M. Hellman, “An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance”, *IEEE Transactions on Information Theory*, **24** (1978), 106-110.
- [32] J. Pollard, “Monte Carlo methods for index computation mod p ”, *Mathematics of Computation*, **32** (1978), 918-924.
- [33] T. Satoh and K. Araki, “Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves”, *Commentarii Mathematici Universitatis Sancti Pauli*, **47** (1998), 81-92.
- [34] I. Semaev, “Evaluation of discrete logarithms in a group of p -torsion points of an elliptic curve in characteristic p ”, *Mathematics of Computation*, **67** (1998), 353-356.
- [35] N. Smart, “The discrete logarithm problem on elliptic curves of trace one”, preprint, 1997.
- [36] P. van Oorschot and M. Wiener, “On Diffie-Hellman key agreement with short exponents”, *Advances in Cryptology – Eurocrypt '96*, Lecture Notes in Computer Science, **1070**, Springer-Verlag, 1996, 332-343.
- [37] Y. Yacobi, “A key distribution paradox”, *Advances in Cryptology – Crypto '90*, Lecture Notes in Computer Science, **537**, Springer-Verlag, 1991, 268-273.